**Федеральное государственное автономное образовательное учреждение высшего образования «Национальный исследовательский университет ИТМО»**

**Факультет программной инженерии и компьютерной техники**

**Вычислительная математика**

**Лабораторная работа №4**

**Вариант 2**

Группа: P3266

Выполнила:

Бореева В. Ю.  
Тихонов И. В.

Проверил:

Машина Е. А.

Г. Санкт-Петербург

2024

# Цель работы

Разобраться с понятием аппроксимации

# 1 Вычислительная часть задачи:

1. Сформировать таблицу табулирования заданной функции на

указанном интервале

Изображение выглядит как текст, снимок экрана, число, Шрифт

Автоматически созданное описание

Изображение выглядит как текст, снимок экрана, число, Шрифт

Автоматически созданное описание

Изображение выглядит как текст, снимок экрана, число, Шрифт

Автоматически созданное описание

2. Построить линейное и квадратичное приближения по 11 точкам

заданного интервала;

Изображение выглядит как текст, снимок экрана, число, Шрифт

Автоматически созданное описание

Изображение выглядит как текст, снимок экрана, число, Шрифт

Автоматически созданное описание

Изображение выглядит как текст, снимок экрана, число, Шрифт

Автоматически созданное описание

Изображение выглядит как текст, программное обеспечение, число, снимок экрана

Автоматически созданное описание

Изображение выглядит как текст, число, Шрифт, программное обеспечение

Автоматически созданное описание

Изображение выглядит как текст, число, программное обеспечение, снимок экрана

Автоматически созданное описание

Изображение выглядит как текст, снимок экрана, число, Шрифт

Автоматически созданное описание

Изображение выглядит как текст, снимок экрана, число, Шрифт

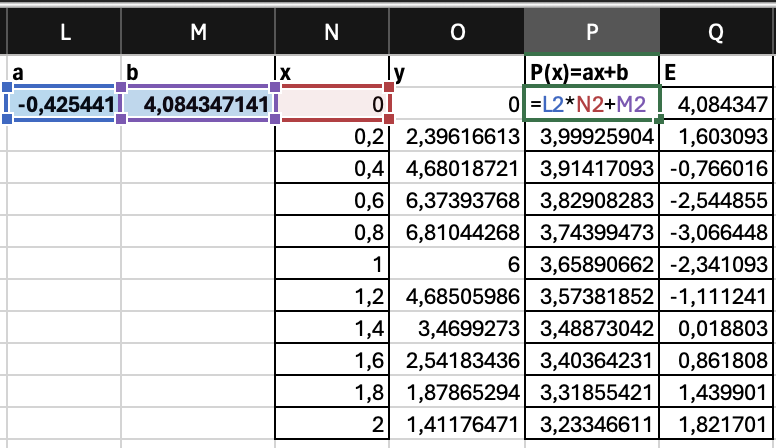
Автоматически созданное описание

Изображение выглядит как текст, число, Шрифт, снимок экрана

Автоматически созданное описание

Изображение выглядит как текст, число, Шрифт, снимок экрана

Автоматически созданное описание



Изображение выглядит как текст, снимок экрана, Шрифт, число

Автоматически созданное описание

Изображение выглядит как текст, снимок экрана, число, Шрифт

Автоматически созданное описание

Изображение выглядит как текст, снимок экрана, число, Шрифт

Автоматически созданное описание

Изображение выглядит как текст, снимок экрана, число, Шрифт

Автоматически созданное описание

Изображение выглядит как текст, снимок экрана, число, Шрифт

Автоматически созданное описание

Изображение выглядит как текст, снимок экрана, Шрифт, линия

Автоматически созданное описание

Изображение выглядит как текст, число, Шрифт, снимок экрана

Автоматически созданное описание

Изображение выглядит как текст, снимок экрана, Шрифт, число

Автоматически созданное описание

Изображение выглядит как текст, чек, Шрифт, документ

Автоматически созданное описание

Изображение выглядит как текст, чек, Шрифт, алгебра

Автоматически созданное описание

Изображение выглядит как текст, чек

Автоматически созданное описание

Изображение выглядит как текст, чек, алгебра

Автоматически созданное описание

Изображение выглядит как текст, Шрифт, белый, снимок экрана

Автоматически созданное описание

Изображение выглядит как текст, снимок экрана, чек, черно-белый

Автоматически созданное описание

Изображение выглядит как текст, снимок экрана, Шрифт, число

Автоматически созданное описание

# 2 Программная часть задачи:

**import** numpy **as** np

**from** Equations **import** \*

**def** exponential(xarr, yarr, n):

# MHK

x = np.array(xarr)

y = np.array(yarr)

coeffs = np.polyfit(x, np.log(y), 1)

a = np.exp(coeffs[1])

b = coeffs[0]

parr = []

earr = []

**for** i **in** range(n):

parr.append(round(a \* pow(math.e, b \* xarr[i]), 4))

earr.append(round(parr[i] - yarr[i], 4))

**return** parr, earr, SKO(parr, yarr, n), determination(parr, yarr, n)

**import** numpy **as** np

**from** Equations **import** \*

**def** exponential(xarr, yarr, n):

# MHK

x = np.array(xarr)

y = np.array(yarr)

coeffs = np.polyfit(x, np.log(y), 1)

a = np.exp(coeffs[1])

b = coeffs[0]

parr = []

earr = []

**for** i **in** range(n):

parr.append(round(a \* pow(math.e, b \* xarr[i]), 4))

earr.append(round(parr[i] - yarr[i], 4))

**return** parr, earr, SKO(parr, yarr, n), determination(parr, yarr, n)

**import** numpy **as** np

**from** Equations **import** \*

**def** logarithmic(xarr, yarr, n):

# MHK

x = np.array(xarr)

y = np.array(yarr)

coeffs = np.polyfit(np.log(x), y, 1)

a = float(coeffs[1])

b = float(coeffs[0])

parr = []

earr = []

**for** i **in** range(n):

parr.append(round(a \* math.log(xarr[i]) + b, 4))

earr.append(round(parr[i] - yarr[i], 4))

**return** parr, earr, SKO(parr, yarr, n), determination(parr, yarr, n)

**import** numpy **as** np

**from** Equations **import** \*

**def** logarithmic(xarr, yarr, n):

# MHK

x = np.array(xarr)

y = np.array(yarr)

coeffs = np.polyfit(np.log(x), y, 1)

a = float(coeffs[1])

b = float(coeffs[0])

parr = []

earr = []

**for** i **in** range(n):

parr.append(round(a \* math.log(xarr[i]) + b, 4))

earr.append(round(parr[i] - yarr[i], 4))

**return** parr, earr, SKO(parr, yarr, n), determination(parr, yarr, n)

**import** numpy **as** np

**from** Equations **import** \*

**def** logarithmic(xarr, yarr, n):

# MHK

x = np.array(xarr)

y = np.array(yarr)

coeffs = np.polyfit(np.log(x), y, 1)

a = float(coeffs[1])

b = float(coeffs[0])

parr = []

earr = []

**for** i **in** range(n):

parr.append(round(a \* math.log(xarr[i]) + b, 4))

earr.append(round(parr[i] - yarr[i], 4))

**return** parr, earr, SKO(parr, yarr, n), determination(parr, yarr, n)

**import** numpy **as** np

**from** Equations **import** \*

**def** logarithmic(xarr, yarr, n):

# MHK

x = np.array(xarr)

y = np.array(yarr)

coeffs = np.polyfit(np.log(x), y, 1)

a = float(coeffs[1])

b = float(coeffs[0])

parr = []

earr = []

**for** i **in** range(n):

parr.append(round(a \* math.log(xarr[i]) + b, 4))

earr.append(round(parr[i] - yarr[i], 4))

**return** parr, earr, SKO(parr, yarr, n), determination(parr, yarr, n)

**import** numpy **as** np

**from** Equations **import** \*

**def** logarithmic(xarr, yarr, n):

# MHK

x = np.array(xarr)

y = np.array(yarr)

coeffs = np.polyfit(np.log(x), y, 1)

a = float(coeffs[1])

b = float(coeffs[0])

parr = []

earr = []

**for** i **in** range(n):

parr.append(round(a \* math.log(xarr[i]) + b, 4))

earr.append(round(parr[i] - yarr[i], 4))

**return** parr, earr, SKO(parr, yarr, n), determination(parr, yarr, n)

**import** numpy **as** np

**from** Equations **import** \*

**def** logarithmic(xarr, yarr, n):

# MHK

x = np.array(xarr)

y = np.array(yarr)

coeffs = np.polyfit(np.log(x), y, 1)

a = float(coeffs[1])

b = float(coeffs[0])

parr = []

earr = []

**for** i **in** range(n):

parr.append(round(a \* math.log(xarr[i]) + b, 4))

earr.append(round(parr[i] - yarr[i], 4))

**return** parr, earr, SKO(parr, yarr, n), determination(parr, yarr, n)

ВЫВОД

В ходе выполнения лабораторной работы мы разобрались с понятием аппроксимации